Accessing the Intel® Random Number Generator through a CSP for Microsoft* CryptoAPI

Intel Platform Security Division

Contents

Introduction

Intel Corporation's vision for computing is "a billion connected PCs, a million connected servers." Security in computing is a fundamental requirement to make this happen. Consequently, Intel believes the industry should work together to develop the "Trusted, Connected PC." The Trusted, Connected PC will enhance today's PCs, servers, and workstations with security capabilities that enable users to perform more secure transactions and information exchange over the Internet with privacy, and without fear of theft, viruses, or other attacks.

Developing a hardware foundation for security is one key element to making Intel's security vision a reality. By implementing key security features within hardware for every computing platform, enhanced security will become ubiquitous. A security solution implemented in hardware is often more robust than a software solution, since hardware has the unique ability to hide the execution and storage of information. Better security solutions implemented across all systems will pave the way towards increased connectivity, access to new products and services, and new business models.

Scope of this Document

This paper details how applications can access Intel's platform silicon security features through the Intel® Hardware Cryptographic Service Provider (CSP) based on the Microsoft* Cryptographic Service Provider Developer's Kit (MS-CSPDK). Included is a brief overview of the first Intel® platform silicon feature, the Intel® Random Number Generator (RNG). The paper details how applications can access the Intel® RNG security feature through Microsoft's Cryptographic Application Programming Interface (CryptoAPI), providing applications with an easy migration path from existing software-based solutions. The Intel Hardware CSP supports Microsoft's Windows* 95, 98, NT4, and Windows 2000 operating systems.

* All other brands and names are the property of their respective owners.

Intel® Random Number Generator Overview

The Intel Random Number Generator is a fundamental building block for improving the ability of computers to preserve the confidentiality of electronic communications. Random number generation is a key component of the encryption algorithms that protect data. Most random number generators available on systems today are software RNGs (often called pseudo RNGs, or PRNGs), which are not capable of generating truly random data. Because pseudo RNGs generate random data by means of a fixed algorithm, their output can be predictable. This predictability weakens software-only encryption schemes relative to hardware-based systems.

Intel's silicon-based random number generator generates true[1] random numbers (numbers which are unpredictable and non-repeating), which can increase the strength of an encryption system. The Intel RNG passes all FIPS 140-1[2] RNG statistical tests, making it a preferred solution wherever random numbers are required. The Intel® hardware random number generator is a key component for use in any strong security solution.

Businesses and consumers rely on networks for communication, using the PC as their protected entry point. In a world that increasingly depends on digital information, PC users can now justifiably expect more security. The silicon-based Intel RNG provides a strong foundation for computer data security. Intel's hardware-based security building blocks enable OEMs to deliver new security technology on broadly deployed IA (Intel® Architecture) platforms.

---

[1] Some argue that it is not possible to generate a "true" random number. This paper assumes Schneier's definition of true (or real) random number generators—they generate sequences that look random, are unpredictable, and cannot be reliably reproduced [4]. More detailed discussions of "true" versus "pseudo-" random numbers are presented in [1], [2], and [3].

[2] FIPS is the United States government's Federal Information Processing Standard. A publication of the National Institute of Standards and Technology, the FIPS 140-1 specification describes government requirements for cryptographic modules for sensitive, but unclassified use. For general information about FIPS 140, see the FAQ from Corsec Security, Inc., at http://www.corsec.com/FIPS140-1FAQ.html, or the FIPS 140-1 publication [5].

Introduction to Microsoft's Cryptographic Application Programming Interface

The Microsoft Cryptographic Application Programming Interface (CryptoAPI), provides services that enable application developers to add authentication, encoding, and encryption to their Microsoft Win32*-based applications. The Intel Hardware Cryptographic Service Provider plug-in for CryptoAPI 2.0 is a CryptoAPI-based Cryptographic Service Provider (CSP) that provides access to the Intel Random Number Generator.

Designed for Microsoft CryptoAPI version 2.0, the Intel® Security CSP is of type PROV_INTEL_SEC. Its primary capability is to provide access to the Intel Random Number Generator and produce hardware-based random numbers. The CSP does not provide any data encryption and decryption, authentication, signing, or key management functions and, thus, can be exported to other countries. The Intel Hardware Cryptographic Service Provider is distributed with the Intel® Security Driver version 1.0 and provides the following features:

- Application accessible CSP information (name, type, and version number).
- Detection and initialization of the Intel RNG during CSP initialization.
- Usages of the Intel RNG to return random numbers through the CryptoAPI.
- Verification of the CSP signature every time the Intel® CSP is loaded to ensure integrity.

Using the Intel® Hardware Cryptographic Service Provider

While the CSP contains a framework of all CryptoAPI interface functions, only those functions that are directly applicable to the functionality of the Intel Security Driver have been implemented. This includes functions for getting CSP information, and retrieving hardware-generated random numbers. An application can get the CSP's information through the CryptGetProvParam function with or without the Intel security hardware present. The CSP returns a failure status for any hardware failure.

The Intel Hardware CSP supports the following Microsoft CryptoAPI functions: CryptAcquireContext, CryptReleaseContext, CryptGetProvParam, CryptSetProvider, and CryptGetRandom. All other Microsoft CryptoAPI functions are not supported by the Intel Hardware CSP, and return a result of E_NOTIMPL. The CSP is installed as the default CSP of type PROV_INTEL_SEC. The header file, icsp4ms.h in the \inc directory, includes the defined value for the PROV_INTEL_SEC type. Any application that requires the Intel Hardware CSP's services can reference the PROV_INTEL_SEC type by either including the icsp4ms.h file or directly using the type value.

Installing the CSP

The Intel Hardware CSP files are installed with the Intel Security Driver as described below.

Step 1 — File Copy

The installation program copies the CSP DLL to the <*system directory*> and copies the icsp4ms.h header file to <*drive*>**:**<*path*>**\Sample Applications\Source\inc**

Where,
<*system directory*> is:
\Windows\System if Windows 95 or Windows 98
\WINNT\System32 if Windows NT* 4.0 or Windows 2000.

<*drive*>**:**<*path*> is **C:\Program Files\Intel\Intel Security Driver**, by default.

Step 2 — Registry Modification

The final requirement for Setup will be to make the appropriate registry changes to make the Intel CSP the Default Provider for its type and copy the CSP signature, which will be verified by the host operating system. The changes made to the registry are:

- Registering the CSP
  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Defaults\Provider
  >Name:REG_SZ: *Intel Hardware Cryptographic Service Provider*
  >Image Path:REG_SZ:*iCSP.DLL*
  >Signature:REG_BINARY:*<digital signature>*
  >Type:REG_DWORD: 0x00000*15*

- Setting the Machine Default CSP
  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Defaults\Provider Types\Type 022
  >Name:REG_SZ: *Intel Hardware Cryptographic Service Provider*

- Setting the User Default CSP
  HKEY_CURRENT_USER\Software\Microsoft\Cryptography\Providers\Type 22
  >Name:REG_SZ: *Intel Hardware Cryptographic Service Provider*

Accessing the Intel Random Number Generator through Microsoft's CryptoAPI

The CSP is the default PROV_INTEL_SEC type CSP. The icsp4ms.h in the \inc directory includes the defined value for the PROV_INTEL_SEC type. Any application that requires the Intel Security CSP's services can reference the PROV_INTEL_SEC type by either including the icsp4ms.h file or directly using the type value.

An application first needs to get a handle to the Intel hardware CSP through the CryptAcquireContext function call. The provider handle can then be used to get CSP information or request a HW random number from the Intel Hardware CSP. When you connect to the Intel Hardware CSP via this CryptAcquireContext function call, it is always a good practice to specify the CSP name (`"Intel Hardware Cryptographic Service Provider"`) using the pre-defined constant INTEL_DEF_PROV. This indicates that you are indeed communicating with the Intel Hardware CSP instead of the default full provider CSP.

An application can get the CSP's information through the CryptGetProvParam function with or without Intel's security hardware present. The information returned includes: CSP name, provider type, and version number.

The pseudo-code below demonstrates the process for accessing the CSP.

```
//Pseudo-code example of acquiring the Intel(R) Hardware CSP

#define WIN32_WINNT 0x400
#include <wincrypt.h>
#include icsp4ms.h

// The CSP provider name INTEL_DEF_PROV is defined in the icsp4ms.h file as
// "Intel Hardware Cryptographic Service Provider
HCRYPTPROV    hProv;
DWORD         version = 0;
DWORD          len = 0;
// Get a handle to the Intel CSP.
If(!CryptAcquireContext(&hProv, NULL, INTEL_DEF_PROV, PROV_INTEL_SEC, 0))
{    printf("Error %x during CryptAcquireContext!\n", GetLastError());    return;}
// Get length of the CSP version and get the CSP version
if (CryptGetProvParam(hProv, PP_VERSION, NULL, &len, 0) == TRUE)
{
   if (CryptGetProvParam(hProv, PP_VERSION, (BYTE*)&version, &len, 0) == TRUE)
   {
       // Compare the version (>= 1.0)
   }
}

// Release the CSP handle
CryptReleaseContext(hProv, 0);
```

Generating Random Numbers

The Intel RNG call can be made through the CryptoAPI CryptGetRandom function to retrieve random numbers from the Intel Security Driver. Once the caller obtains the provider handle through the CryptAcquireContext function, the handle can be used to retrieve the HW RNG. The CSP returns fail status if the HW RNG is not available or enabled on the platform. It returns CRYPT_SUCCEED if the operation succeeded. If the operation fails, one of the following error codes may be set:

- ERROR_INVALID_PARAMETER: If the caller does not provide a buffer.
- ERROR_DEV_NOT_EXIST: The Intel RNG HW is not available or no longer exists.

The following pseudo-code demonstrates how to generate a random number.

```
//Pseudo-code example of how to generate a random number
#define WIN32_WINNT 0x400
#include <wincrypt.h>
#include icsp4ms.h

// The CSP provider name INTEL_DEF_PROV is defined in the icsp4ms.h file as
// "Intel Hardware Cryptographic Service Provider
HCRYPTPROV    hProv;
DWORD         randomNumber = 0;
DWORD         randomLength = 4;

// Get a handle to the Intel CSP.
If(!CryptAcquireContext(&hProv, NULL, INTEL_DEF_PROV, PROV_INTEL_SEC, 0))
{    printf("Error %x during CryptAcquireContext!\n", GetLastError());    return FAIL;}
// Get 4-bytes random number
if (CryptGenRandom(hProv, randomLength, (BYTE*)&randomNumber) != TRUE)
{
    // Release the CSP handle
    CryptReleaseContext(hProv, 0);
    return FAIL;
}

// Release the CSP handle
CryptReleaseContext(hProv, 0);
//Operate on random data
```

Conclusion

Random numbers are the foundation of secure cryptographic solutions, digital signatures, and protected communications protocols. The Intel RNG provides a high quality, hardware-based random number generator suitable as a fundamental security building block. Existing CryptoAPI-based applications can easily upgrade current random number usage to the Intel RNG with minor changes to the application's existing code base. Easy access to the Intel RNG through the CryptoAPI security framework protects existing CryptoAPI development efforts by ISVs.

References

[1] Davies, Robert. "True Random Numbers." http://webnz.com/robert/true_rng.html (9 Oct. 1998).

[2] "Diehard." http://stat.fsu.edu/~geo/diehard.html (16 Oct. 1998).

[3] Ellison, Carl. "Cryptographic Random Numbers." Draft P1363 Appendix E. http://www.clark.net/pub/cme/P1363/ranno.html (9 Oct. 1998).

[4] Schneier, Bruce. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. New York: John Wiley & Sons, 1996.

[5] FIPS 140-1, "Security Requirements for Cryptographic Modules." Federal Information Processing Standards Publication 140-1. U.S. Department of Commerce/NIST, National Technical Information Service. Springfield, Virginia, 1994. http://csrc.ncsl.nist.gov/fips/fips1401.htm (16 Oct. 1998).